## ***REMARKS***

Claim 18-37 are pending in this application. The Examiner has objected to dependent claims 34-37 as improper because the preambles are not written in a proper dependent form. Applicants have corrected the preambles of dependent claims 34-37 according to the Examiner's suggestions. The Examiner has also rejected claims 18-37 under 35 U.S.C. §102(e) as being anticipated by U.S. Patent Application No. US2002/0100029 ("*Bowen*") filed by Matt Bowen. Applicants respectfully traverse this rejection and request reconsideration and allowance of claims 18-37 for the following reasons.

Applicant respectfully submits that *Bowen* neither discloses nor suggests the pending claims. According to M.P.E.P §706.02, "for anticipation under 35 U.S.C. 102, the reference must teach *every aspect* of the claimed invention..." (emphasis added). However, *Bowen* fails to disclose or suggest various elements of the claimed invention.

The Examiner states that the parser/parsing step of claims 18, 25, and 26 is disclosed in paragraph 113 of the *Bowen* reference. However, that paragraph does not disclose the claimed parser for "parsing the functional description expressed in the interpretive, algorithmic language with ***at least one undeclared variable*** into an abstract syntax tree." (emphasis added). The claim language specifically recites parsing "an algorithmic language containing at least one ***undeclared variable.***" In contrast, the *Bowen* reference discloses a method of parsing and compiling a C language program. C language programs, by definition, cannot contain any undeclared variables. Indeed, the *Bowen* reference clearly indicates that Bowen's invention must receive ***declared*** variables:

> In this embodiment the user writes a description 202 of the system in a C-like language, which is actually ANSI C with some additions which allow efficient translation to

> hardware and parallel processes. This input description will be compiled by the system 200 of FIG. 2. The additions to the ANSI C language include the following: ***Variables are declared with explicit bit widths and the operators working on the variables work with an arbitrary precision. This allows efficient implementation in hardware.*** For instance a statement which declares the width of variables (in this case the program counter pc, the instruction register ir, and the top of stack tos) is as follows: unsigned 12 pc, ir, tos.

*Bowen,* ¶¶61-63 (emphasis added). This description clearly demonstrates that *Bowen* parser only operates on ***declared*** variables. Moreover, nowhere does *Bowen* describe the parsing or processing of ***undeclared variables.*** Therefore, the parser/parsing step for parsing ***at least one undeclared variable*** of claims 18, 25 and 26 is not disclosed by *Bowen.*

Moreover, the Examiner cites pages 19-32 as disclosing the parser/parsing step of the present invention. Pages 19-32 are three appendices comprising source code examples of a target system. Appendix 1 is a register transfer level (RTL) description of a simple processor. *Bowen,* ¶56. Appendix 2 is an RTL description of examples depicted in FIGS. 4 through 6. *Bowen,* ¶57. Appendix 3 is an input specification for a target system. *Bowen,* ¶58. None of these Appendices disclose or describe parsing at all. Additionally, pages 19, 21, and 22 show source code specifically for ***declaring variables***. As described above, the parser/parsing step of the present invention specifically recites the processing and use of ***undeclared variables.*** Therefore, the appendices on pages 19-32 do not describe the parser/parsing steps of the claimed invention.

The Examiner states that the type/shape analyzer and the step assigning the inferred type and dimension to the undeclared variable of claims 18, 25, and 26 are disclosed in Figure 9 of *Bowen.* As described above, *Bowen* does not disclose the processing of ***undeclared variables*** or the inferring of type and shape to undeclared variables. Rather, Figure 9 is described in *Bowen* as a dependency tree - a graphical representation of how the variables depend on one another. The only description in *Bowen* of Figure 9 is illustrative:

> The partitioner 208 generates a dependency graph as shown in FIG. 9 which indicates which variables depend on which. It is used by the partitioner to determine the communications costs associated with the partitioning, for instance to assess the need for variables to be passed from one resource to another given a particular partitioning.

*Bowen,* ¶237. There is nothing in *Bowen's* description of Fig. 9 disclosing a type-shape analyzer that infers the type and dimension of an undeclared variable. For example, inferring a type requires determining whether a variable is signed or unsigned, floating point or integer, by analyzing the way the variable is used in the program. Similarly, inferring shape requires, for example, determining whether a variable is a single element, an array of a certain number elements or a multidimensional array of a certain number of elements by analyzing the usage of the variable in the program. In contrast, the description of Fig. 9 indicates that it is used to merely to determine how variables pass from one resource to another. A variable's type and dimension are not used in determining what variables are needed by various resources. Thus, Fig. 9 has nothing to do with "inferring a type and a dimension to an undeclared variable" as recited by claims 18, 25 and 26. Similarly, Fig. 9 does not describe or illustrate "assigning and inferred type and dimension to an undeclared variable." Therefore, these claim elements are not disclosed by *Bowen.*

The Examiner states that *Bowen* discloses the "statement deconstructor" as the block labeled "210" of Fig. 2. Block 210 of Fig. 2 is labeled as an "Interface Cosynthesizer" and is depicted at a split between the "hardware" and "software" branches of the design flow for a hybrid hardware/software implementation. As described below by *Bowen,* the Interface Cosynthesizer creates the interface between functions that will be performed in hardware versus those functions that will be performed in software in the final design. Below are the only two descriptions of the Interface Cosynthesizer from *Bowen*:

- "Interfaces between the hardware and software are instantiated by the interface cosynthesizer 210 from a standard library of available communication mechanisms." *Bowen,* ¶132.
- "The interface cosynthesizer 210 implements the interface between hardware and software on two FPGA channels 804 and 808 and these are used to transfer a position information to the rendering process and to transfer the button information to the position calculation 806 from button input 822." *Bowen,* ¶234.

Neither of these descriptions discloses the statement deconstructor of the present invention. The claims recite a statement desconstructor "for transforming a compound statement in the abstract syntax tree into at least one simple statement." Transforming a compound statement into individual statements is not the same as providing the interface details between functions implemented in hardware versus software as described by *Bowen.* Furthermore, transforming a compound statement into a series of single statements is independent of whether the function is preformed in hardware or software. Therefore, *Bowen's* Interface Cosynthesizer does not disclose the statement deconstructor of the present invention.

The Examiner states that the "translator…for translating the abstract syntax tree into a register level format" of the claimed invention is disclosed by *Bowen's* "RTL Synthesis" block depicted as box 214 of Fig. 2. However, the RTL Synthesis block does not translate an abstract syntax tree into a register level format, i.e. RTL, as required by the claims. Rather, *Bowen's* RTL Synthesis block receives RTL directly, thus no translation from an abstract syntax tree occurs: "[A]n RTL description…can be output to a RTL synthesis system 214 using a hardware description language (e.g. Handel-C or VHDL), or else synthesized to a gate level description using the techniques of De Michelli." *Bowen,* ¶144. Moreover, Fig. 2 clearly shows that the input to the RTL Synthesis block is an "RTL description." Therefore, the translator claimed in the present invention is not same as *Bowen's* RTL Synthesis.

The Examiner states that claim 33 is also anticipated by *Bowen* for similar reasons as already discussed. The Examiner states that the parsing step of claim 33 is disclosed by block 204 and described by paragraph 113. As stated above, the description of block 204 in *Bowen* clearly indicates that all variables are declared and as such is not the same as "parsing the functional description expressed in the interpretive, algorithmic language with at least one undeclared variable into an abstract syntax tree" as recited by claim 33. The Examiner also states that inferring and assigning a type and dimension to the undeclared variable is disclosed in Fig. 9 of *Bowen.* As described above, Fig. 9 is a dependency tree, which neither infers nor assigns a type or shape to variables at all. Additionally, the Examiner states that "transforming compound statements in the abstract syntax tree into a series of single statements" step of claim 33 is performed by block 210 of Fig. 2. Also described above, *Bowen's* Interface Cosynthesizer (210) does not transform compound statements into single statements, but rather, provides the interface details between the functions assigned to hardware and those assigned to software. Finally, the Examiner states that the RTL Synthesis block of Fig 2 discloses the step of claim 33 of "translating the abstract syntax tree into a register transfer level format". However, as discussed above, the RTL Synthesis block receives a register transfer level (RTL) description, and thus does not translate an abstract syntax tree into RTL. Therefore, for the same reasons discussed above for claims 18 and 26, claim 33 is not anticipated by *Bowen.*

Claims 19-25 depend from 18, claims 27-32 depend from 26, and claims 34-37 depend from claim 33. As discussed above, claims 18, 26 and 33 are valid, and as such, claims 19-25, 27-32, and 34-37 are also valid for at least the same reasons. Moreover, these dependent claims are valid for additional reasons. For example, claims 20 and 35 recite "analyzing the value range of the at least one undeclared variable; and assigning the required precision," and claim 28 recites "a precision analyzer." As discussed above, *Bowen* does not disclose or teach a system that analyzes undeclared variables to infer attributes such as type, dimension, and precision. As such, *Bowen* does not anticipate the dependent claims.

## **Conclusion**

In sum, Applicants respectfully submit that claims 18-37 as presented herein, are patentably distinguishable over the cited references (including references cited, but not applied). Therefore, Applicants request reconsideration and allowance of these claims.

Applicants respectfully invite Examiner to contact Applicants' representative at the number provided below if Examiner believes it will help expedite furtherance of this application.

RESPECTFULLY SUBMITTED,
Prithviraj Banerjee, Alok Choudhary, Malay Haldar, Anshuman Nayak

Date: 11/22/04

By: [illegible]
Deepti Panchawagh-Jain, Esq.
Registration No. 43,846
3039 Calle De Las Estrella
San Jose, CA 95148
(408) 274-3043 (Phone)
(801) 838-1085 (Fax)